REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE (DD-MM-YYYY) 15-04-2002	**2. REPORT DATE** FINAL	**3. DATES COVERED:** *(From – To)* 10/01/96-03/31/01
4. TITLE AND SUBTITLE Hardware Verification Integrating Deductive with Algorithmic Technologies		**5a. CONTRACT NUMBER** DABT63-96-C-0096
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Zohar Manna		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stanford University 651 Serra Street **Stanford, CA 94305**		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dept of the Army Directorate of Contracting ATTN: ATZS-DKO-1 P.O. Box 12748 Ft Huachuca, AZ 85670-2748		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

20020423 181

14. ABSTRACT
The work supported under this grant can be divided into four closely related areas:
(1) Verification of real-time and hybrid systems;
(2) Static analysis;
(3) Automata-based deductive verification of real-time systems;
(4) Abstraction and modularity in deductive verification.
The results obtained in these areas have been reported in conference and journal papers referenced in the report. Most of the methods developed have been implemented and evaluated for their utility in the framework of the Stanford Temporal Prover (STeP).

15. SUBJECT TERMS
verification, real-time systems, hybrid systems, abstraction, modularity, invariant generation.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Zohar Manna
a. REPORT UNCLASS	**b. ABSTRACT** UNCLASS	**C. THIS PAGE** UNCLASS	UNLIMITED		**20b. TELEPHONE NUMBER (include area code)** 650 723 4364

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI-Std Z39-18

Hardware Verification: Integrating Deductive with Algorithmic Technologies

R & D Final Report
DARPA Contract DABT63-96-C-0096

P.I.: Prof. Zohar Manna
Computer Science Department
Stanford University
Stanford, CA. 94305-9045

April 15, 2002

1 Introduction

The work supported under this grant can be divided into four closely related areas:

- Verification of real-time and hybrid systems;
- Static analysis;
- Automata-based deductive verification of reactive systems;
- Abstraction and modularity in deductive verification;

In parallel with the theoretical work we designed and implemented the Stanford Temporal Prover (STeP), a comprehensive verification tool, which provided a testbed to evaluate the utility of the models and methods developed.

In the following sections we elaborate on each of the four areas of theoretical work and briefly describe our verification tool. All sections provide references to conference and journal papers describing the results obtained in more detail.

2 Verification of real-time and hybrid systems

2.1 Computational Model and Verification Rules

This work involves the development of a new computational model for real-time and hybrid systems, called the *clocked transition system* (CTS), *phase transition system* model respectively. The CTS model is a development of our previous *timed transition* model, where some of the changes are inspired by the model of *timed automata.* The new model leads to a simpler style of temporal specification and verification, requiring no extension of the temporal language.

We developed verification rules for proving safety properties (including time-bounded response properties) of clocked transition systems, and separate rules for proving (time-unbounded) response properties. All rules are associated with verification diagrams. The verification of *response* properties requires adjustments of the proof rules developed for untimed systems, reflecting the fact that progress in the real time systems is ensured by the progress of time and not by fairness. The style of the verification rules is very close to the verification style of untimed systems which allows the (re)use of verification methods and tools, developed for untimed reactive systems, for proving all interesting properties of real-time systems.

It is important that real-time/hybrid system descriptions are *non-Zeno*, that is, all run prefixes can be extended to an infinite, time-divergent run, because verification results of *Zeno* systems may be misleading. We developed methods to prove that real-time/hybrid systems are *non-Zeno.*

This work has been reported in [KMP98] and [KMP00].

2.2 Invariant Generation and Implementation

We investigated the feasibility of computer-aided deductive verification of hybrid systems. Hybrid systems are modeled by phase transition systems, in which activities specify the bounds on the derivatives of the continuous variables. We developed a method for invariant generation based on static analysis of the phase transition system. The invariants produced can be used as auxiliary properties in the verification of temporal properties. We demonstrated that in some cases the invariants thus produced suffice to prove the main safety property. The methods for invariant generation and the verification rules described above were implemented in STeP (Stanford Temporal Prover). This work is reported in [MS98].

We developed a modular framework for proving temporal properties of real-time systems, based on clocked transition systems and linear-time temporal logic. We demonstrated how deductive verification rules, verification diagrams, and automatic invariant generation can be used to establish properties of real-time systems in this framework. We proposed both global and modular proofs of the branching-time property of non-Zenoness. As a case study, we mechanically verified the *generalized railroad crossing* using the Stanford Temporal Prover, STeP. This work was reported in [BMSU97] and [BMSU01].

3 Static Analysis

3.1 Invariant Generation

Verifying temporal specifications of reactive and concurrent systems relies on generating auxiliary assertions and on strengthening given properties of the system. This can be achieved by two dual approaches. The *bottom-up* method performs an abstract forward propagation (computation) of the system, generating auxiliary assertions; the *top-down* method performs an abstract backward propagation to strengthen given properties. *Exact* application of these methods is complete but is usually infeasible for large-scale verification. *Approximate analysis* techniques are often needed to complete the verification.

Our approach to the generation of invariants and intermediary assertions was to reduce a verification problem to domains that admit specialized and efficient solvers. We extended known methods for generation of auxiliary invariants by formalizing and analyzing a general verification rule that uses *assertion graphs* to generate auxiliary assertions for the verification of general safety formulas. We applied and developed abstract interpretation techniques to find approximations of least and greatest fixed points in the forward and backward analysis of systems. We developed novel methodologies for abstract interpretation of parameterized programs. This work has been reported in [BBM97].

3.2 Generation of Ranking Functions

Deductive verification of progress properties relies on finding ranking functions to prove termination of program cycles. We developed an algorithm to synthesize linear ranking functions that can establish such termination. Fundamental to our approach is the representation of systems of linear inequalities and sets of linear expressions as polyhedral cones. This representation allows us to reduce the search for linear ranking functions to the computation of polars, intersections and projections of polyhedral cones, problems which have well-known solutions. This work has been reported in [CS01].

4 Automata-based Deductive Verification

We developed diagram-based formal methods for verifying temporal properties of finite- and infinite-state reactive systems. These methods, which share a common background and tools, differ in the way they use automatic procedures within an interactive setting based on deduction. They can be used to produce a static proof object, or to perform incremental analysis of systems and specifications. An overview of the diagram-based methods developed in our group was given in [dAMSU97].

4.1 Generalized Verification Diagrams

We developed *generalized Verification Diagrams*, which combine deductive and algorithmic verification to establish general temporal properties of finite- and infinite-state reactive systems. The diagram serves as an abstraction of the system. This abstraction is deductively justified and algorithmically model checked. We developed a new simple class of verification diagrams, using Müller acceptance conditions, and demonstrated how they can be used to verify general temporal properties of reactive systems. This work was reported in [MBSU98].

4.2 Hybrid Diagrams

We developed a methodology for the verification of temporal properties of hybrid systems. The methodology is based on the deductive transformation of *hybrid diagrams*, which represent the system and its properties, and which can be algorithmically checked against the specification. This check either gives a positive answer to the verification problem, or provides guidance for the further transformation of the diagrams. The resulting methodology is complete for quantifier-free linear-time temporal logic. This work was reported in [dAKM97].

4.3 Deductive Model Checking

We developed Deductive Model Checking: an extension of classical tableau-based model checking procedures to the case of infinite-state systems, using deductive methods in an incremental construction of the behavior graph. Logical formulas are used to represent infinite sets of states in an abstraction of this graph, which is repeatedly refined in the search for a counterexample computation, ruling out large portions of the graph before they are expanded to the state-level. This can lead to large savings, even in the case of finite-state systems. Only local conditions need to be checked at each step, and previously proven properties can be used to further constrain the search. Although the resulting method is not always automatic, it provides a flexible, general and complete framework that can integrate a diverse number of other verification tools. The work on Deductive Model Checking has been reported in [SUM99].

4.4 Verification of Parameterized Systems

We developed a visual approach to proving progress properties of parameterized systems using induction on verification diagrams. The inductive hypothesis is represented by an automaton and is based on a state-dependent order on process indices, for increased flexibility. This approach yields more intuitive proofs for progress properties and simpler verification conditions that are more likely to be proved automatically. This work has been reported in [MS99].

4.5 Generalizing Verification Rules

We developed a method based on alternating automata on infinite words to reduce the verification of linear temporal logic LTL safety properties over infinite-state systems to the proof of first-order verification conditions. This method generalizes the traditional deductive verification approach of providing verification rules for particular classes of formulas, such as invariances, nested precedence formulas, etc. It facilitates the deductive verification of arbitrary safety properties without the need for explicit temporal reasoning. This work was reported in [MS00].

5 Abstraction and Modularity

5.1 Modular Verification Framework

We developed a formal framework for the modular description and verification of parameterized fair transition systems. The framework allows us to apply existing global verification methods, such as verification rules and diagrams, in a modular setting. Transition systems and transition modules can be described by recursive module expressions, allowing the description of hierarchical systems of unbounded depth. Apart from the usual *parallel composition*, *hiding* and *renaming* operations, our module description language provides constructs to *augment* and *restrict* the module interface, capablilities that are essential for recursive descriptions. We developed several deductive proof techniques to establish and re-use modular properties, including a *modular verification rule* to prove properties over modules with non-recursive descriptions; a property *inheritance* mechanism that provides an incremental proof method such that properties of module A can be reused in any module B whose description refers to A; *modular abstraction*, which allows us to focus the proof on relevant components only; and an *induction rule*, which makes the methodology applicable to recursive designs. We demonstrated that our framework allows the use of assumption-guarantee reasoning without suffering from its main disadvantage of having to identify sufficiently strong guarantee properties up front. In a case study we showed that in our framework assumptions are generated naturally in the course of the proof. This work was reported in [FMS98].

5.2 Assertion-based Abstraction

We developed a method to generate finite-state abstractions of reactive systems using decision procedures. The algorithm compositionally abstracts the transitions of the system, relative to a given, fixed set of assertions. With this method, the number of validity checks is proportional to the size of the system description, rather than the size of the abstract state-space. The resulting abstract state-space is finite and can then be explored by a model checker. The procedure provides an alternative method for combining deductive and algorithmic verification. The use of deductive tools makes the procedure applicable to infinite-state systems. However, the efficiency of the abstraction procedure, and the use of finite-state model checking at the abstract level, gives the procedure a level of automation comparable to that of finite-state algorithmic methods. This work was reported in [CU98].

5.3 Combining Modularity and Abstraction

When designing a system modularly, one would like to prove simple properties of individual modules, and combinations of modules, before the entire system is specified. *Assumption-guarantee reasoning* is often used to prove properties of a module that depend on its environment, before that environment is fully specified. Abstraction can facilitate this process: Modular properties can be model checked for abstractions, relative to assumptions on the environment. Furthermore, for real-time and hybrid systems, part or all of the complex real-time behavior can be abstracted away when debugging individual modules. More expensive verification methods should only be used after the design components and some of their combinations pass these simple (and fast) checks. We successfully applied a combination of modular and abstraction techniques to the *steam boiler*, a benchmark case study in specification and verification methods for hybrid controlled systems. A detailed description of this work appeared in [MCF$^+$97].

6 Tool: Stanford Temporal Prover

We designed and implemented STeP (Stanford Temporal Prover), a comprehensive verification tool for the verification of temporal properties of reactive, real-time and hybrid systems, including parameterized systems. It provides deductive, algorithmic, and deductive-algorithmic verification methods. A powerful validity checker, which incorporates an integration of decision procedures for most of the theories common in verification conditions [Bjø98] is available to automatically discharge most of the resulting first-order verification conditions.

6.1 Scope

Most of the theoretical work described above has been implemented in STeP including support for real-time [BMSU01] and hybrid systems [MS98], invariant generation [BBM97], automatic generation of ranking functions [CS01], a graphical user interface to draw diagrams and automatic generation of diagram verification conditions [MBSU98], assertion-based abstraction [CU98], generalized verification rules based on alternating automata [MS00], and support for modularity [FMS98], thus providing us with valuable feedback on the utility of our methods in practice.

6.2 Case studies

Several case studies have been performed with STeP, including the deductive verification of a parameterized fault-tolerant protocol, which settled the correctness for the general N-process case, where previously only small instances of the protocol had been verified [BLM97], the steamboiler benchmark system [MCF+97], and an unbounded, recursively defined arbiter that guarantees mutual exclusion to a critical resource [FMS98].

6.3 Implementation

The original version of STeP [BBC+95] was implemented in ML. The current version of STeP [BBC+00] is implemented in Java, except for the first-order theorem prover and validity checker, which are still in ML.

7 Dissertations

The following dissertations were in part supported by this grant:

- Nikolaj S. Bjørner, Integrating Decision Procedures for Temporal Verification, 1998 [Bjø98].
- Tomás E. Uribe, Abstraction-based Deductive-Algorithmic Verification of Reactive Systems, 1998 [Uri98].
- Henny B. Sipma, Diagram-based Verification of Discrete, Real-time and Hybrid Systems, 1999 [Sip99].

References

[BBC+95] N.S. Bjørner, A. Browne, E.S. Chang, M. Colón, A. Kapur, Z. Manna, H.B. Sipma, and T.E. Uribe. STeP: The Stanford Temporal Prover, User's Manual. Technical Report STAN-CS-TR-95-1562, Computer Science Department, Stanford University, November 1995. available from `http://www-step.stanford.edu/`.

[BBC+00] N.S. Bjørner, A. Browne, M. Colón, B. Finkbeiner, Z. Manna, H.B. Sipma, and T.E. Uribe. Verifying temporal properties of reactive systems: A STeP tutorial. *Formal Methods in System Design*, 16(3):227–270, June 2000.

[BBM97] N.S. Bjørner, A. Browne, and Z. Manna. Automatic generation of invariants and intermediate assertions. *Theoretical Computer Science*, 173(1):49–87, February 1997.

[Bjø98] N.S. Bjørner. *Integrating Decision Procedures for Temporal Verification.* PhD thesis, Computer Science Department, Stanford University, November 1998.

[BLM97] N.S. Bjørner, U. Lerner, and Z. Manna. Deductive verification of parameterized fault-tolerant systems: A case study. In *Intl. Conf. on Temporal Logic.* Kluwer, 1997. To appear.

[BMSU97] N.S. Bjørner, Z. Manna, H.B. Sipma, and T.E. Uribe. Deductive verification of real-time systems using STeP. In *4th Intl. AMAST Workshop on Real-Time Systems*, vol. 1231 of *LNCS*, pages 22–43. Springer-Verlag, May 1997.

[BMSU01] N.S. Bjørner, Z. Manna, H.B. Sipma, and T.E. Uribe. Deductive verification of real-time systems using STeP. *Theoretical Computer Science*, 253:27–60, 2001.

[CS01] M. Colon and H. Sipma. Synthesis of linear ranking functions. In T. Margaria and W. Yi, editors, *7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, vol. 2031 of *LNCS*, pages 67–81. Springer Verlag, April 2001.

[CU98] M.A. Colón and T.E. Uribe. Generating finite-state abstractions of reactive systems using decision procedures. In A.J. Hu and M.Y. Vardi, editors, *Proc. 10^{th} Intl. Conference on Computer Aided Verification*, vol. 1427 of *LNCS*, pages 293–304. Springer-Verlag, July 1998.

[dAKM97] L. de Alfaro, A. Kapur, and Z. Manna. Hybrid diagrams: A deductive-algorithmic approach to hybrid system verification. In *14th Symposium on Theoretical Aspects of Computer Science*, vol. 1200 of *LNCS*, pages 153–164, February 1997.

[dAMSU97] L. de Alfaro, Z. Manna, H.B. Sipma, and T.E. Uribe. Visual verification of reactive systems. In *3rd Intl. Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, vol. 1217 of *LNCS*, pages 334–350. Springer-Verlag, April 1997.

[FMS98] B. Finkbeiner, Z. Manna, and H.B. Sipma. Deductive verification of modular systems. In W.P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference, COMPOS'97*, vol. 1536 of *LNCS*, pages 239–275. Springer-Verlag, December 1998.

[KMP98] Y. Kesten, Z. Manna, and A. Pnueli. Verification of clocked and hybrid systems. In G. Rozenberg and F.W. Vaandrager, editors, *Lectures on Embedded Systems*, vol. 1494 of *LNCS Tutorial*, pages 4–73. Springer, Heidelberg, 1998.

[KMP00] Y. Kesten, Z. Manna, and A. Pnueli. Verification of clocked and hybrid systems. *Acta Informatica*, 36:837–912, 2000.

[MBSU98] Z. Manna, A. Browne, H.B. Sipma, and T.E. Uribe. Visual abstractions for temporal verification. In A. Haeberer, editor, *Algebraic Methodology and Software Technology (AMAST'98)*, vol. 1548 of *LNCS*, pages 28–41. Springer-Verlag, December 1998.

[MCF$^+$97] Z. Manna, M.A. Colón, B. Finkbeiner, H.B. Sipma, and T.E. Uribe. Abstraction and modular verification of infinite-state reactive systems. In M. Broy and B. Rumpe, editors, *Requirements Targeting Software and Systems Engineering (RTSE)*, vol. 1526 of *LNCS*, pages 273–292. Springer-Verlag, 1997.

[MS98] Z. Manna and H.B. Sipma. Deductive verification of hybrid systems using STeP. In T. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, vol. 1386 of *LNCS*, pages 305–318. Springer-Verlag, April 1998.

[MS99] Z. Manna and H.B. Sipma. Verification of parameterized systems by dynamic induction on diagrams. In N. Halbwachs and D. Peled, editors, *Proc. 11th Intl. Conference on Computer Aided Verification*, vol. 1633 of *LNCS*, pages 25–43, Trento, Italy, July 1999. Springer-Verlag.

[MS00] Z. Manna and H.B. Sipma. Alternating the temporal picture for safety. In U. Montanari, J.D. Rolim, and E. Welzl, editors, *Proc. 27th Intl. Colloq. Aut. Lang. Prog.*, vol. 1853, pages 429–450, Geneva, Switzerland, July 2000. Springer-Verlag.

[Sip99] H.B. Sipma. *Diagram-based Verification of Discrete, Real-time and Hybrid Systems.* PhD thesis, Computer Science Department, Stanford University, February 1999.

[SUM99] H.B. Sipma, T.E. Uribe, and Z. Manna. Deductive model checking. *Formal Methods in System Design*, 15(1):49–74, July 1999.

[Uri98] T.E. Uribe. *Abstraction-based Deductive-Algorithmic Verification of Reactive Systems.* PhD thesis, Computer Science Department, Stanford University, December 1998. Technical Report STAN-CS-TR-99-1618.